Homework 1 – Knight's Tour

EECS 592 – Introduction to Artificial Intelligence

HM-SHEN

September 22, 2017

Contents

1 Description of the Strategy

1.1 General Description of the Searching Algorithm

To search for closed knight's tour, iterative depth-first search algorithm (DFS) is implemented. However, the memory and time requirement of DFS is very huge. Several heuristic algorithms are applied to optimize the searching process.

Staring from the root node ($(2,3)$ in this homework), iterative DFS algorithm pushes all its neighbors into a stack according to a specific order determined by different heuristics. Then, it iteratively `peek()` the top element in the stack, add it to our current path and push all its valid neighbors into the stack. Once we reach a stage where the current node does not have any valid neighbors and closed knight's tour has not been found, we start backtracking. To backtrack from the current node, `remove()` this node from the path, `pop()` it from the top of the stack and start next iteration by peeking the top of the stack. If the next element on top of the stack is the same as the last element in the path, we should directly `pop()` this node from stack and `remove()` from the path because all its neighbors have been explored and no closed knight's tour has been found. This process is executed iterative until the stack is clear.

1.2 Sorting of Different Strategy

The general framework of finding closed Knights' tour has been briefly described above. One of the most important steps in the above algorithm is the ordering of neighbor nodes in stack. Several heuristics has been given in the task sheet. They are briefly described below.

0. Order neighbors of the current node (the last one in path) randomly.
1. Order the neighbors according to their fixed degree.
2. Order the neighbors according to their dynamic degree.
3. Backup when more than one neighbor of the current node has dynamic degree 1.
4. Prioritize neighbor with dynamic degree 1 and discard its siblings.
5. Combine $2, 3, 4$.

1.3 An Extra Strategy

After doing many experiments, choosing to visit nodes lying along the edges and at corners first seems to be a very good strategy because the alternative, visiting nodes in the center of the board first, tends to isolate those nodes at corner. Those nodes are very hard to be accessed (especially when most of the nodes at center are visited). Based on this idea, we can favor those neighbors at corners when they have the same dynamic degree (in strategy 2). To characterize

this, we can measure the manhattan distance between the candidate neighbors and the four corners and choose the one with the smallest distance to any of the four corners.

2 Experiment Summary

2.1 Statistics of strategy 0 to 6